U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE

CLAIM TO CONVENTION PRIORITY UNDER 35 U.S.C. § 119		Docket Number: 10191/2276	Conf. Number: 6708
Application Number: 10/091,057	Filing Date: March 4, 2002	Examiner: Behzad PEIKARI	Art Unit: 2186
Invention Title: METHOD FOR PROTECTING SOFTWARE PROGRAMS FROM INADVERTENT EXECUTION (as amended)		Inventor(s): HURICH et al.	

Address to:
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on:

Date: May 20, 2005

Signature: _____
Jong H. Lee (Reg. No. 36,197)

A claim to the Convention Priority Date pursuant to 35 U.S.C. § 119 of Application No. 101 10 050.7 filed in the German Patent Office on March 2, 2001, was previously made. To complete the claim to the Convention Priority Date, certified copy of the priority application is attached.

Dated: May 20, 2005

By: _____ (by J.L. Reg. no. 36,197)
Richard L. Mayer (Reg. No. 22,490)

KENYON & KENYON
One Broadway
New York, N.Y. 10004
(212) 425-7200 (telephone)
(212) 425-5288 (facsimile)

CUSTOMER NO. 26646
PATENT & TRADEMARK OFFICE

BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 101 10 050.7

Anmeldetag: 2. März 2001

Anmelder/Inhaber: ROBERT BOSCH GMBH, Stuttgart/DE

Bezeichnung: Verfahren zur Absicherung sicherheitskritischer Programmteile vor versehentlicher Ausführung und eine Speichereinrichtung zur Durchführung dieses Verfahrens

IPC: G 06 F 12/14

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 19. Februar 2002
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Weihmayr

A 9161
03/00
EDV-L

CERTIFIED COPY OF
PRIORITY DOCUMENT

BEST AVAILABLE COPY

R. 39828

28.02.01

ROBERT BOSCH GMBH, 70442 Stuttgart

Verfahren zur Absicherung sicherheitskritischer Programmteile vor versehentlicher Ausführung und eine Speichereinrichtung zur Durchführung dieses Verfahrens

Die vorliegende Erfindung betrifft ein Verfahren zur Absicherung sicherheitskritischer Programmteile und eine Speichereinrichtung zur Durchführung dieses Verfahrens.

Stand der Technik

Die beispielsweise in Steuergeräten von Kraftfahrzeugen eingesetzte Software wird aufgrund der ständig anwachsenden Anforderungen immer komplexer. Dies und die zunehmend differenzierteren Randbedingungen führen zu einer Ausprägung verschiedenster Betriebszustände des Softwaresystems.

Problematisch dabei ist, dass diese Betriebszustände teilweise in höchstem Maße inkompatibel zueinander sind. Sind in ein und demselben Speicher Programmteile abgelegt, welche verschiedenen Betriebszuständen zugeordnet sind, kann ein versehentliches Ausführen eines Programmteils im falschen Betriebszustand zu einem sicherheitskritischen Verhalten führen.

Ein Beispiel hierfür ist die Existenz einer Endlosschleife, welche in Erwartung eines Abschaltens von Außen Sinn machen kann, während des normalen Programmablaufs jedoch niemals zur Ausführung kommen darf. Während der Ausführung solcher Programmteile sind grundsätzliche Überwachungsmechanismen, wie beispielsweise ein Hardware oder ein Software-Watchdog, abgeschaltet oder zumindest unwirksam gemacht.

Eine vollständige Vermeidung der Existenz solcher Programmstrukturen kann heute nicht mehr garantiert werden. Daher müssen diese Programmteile praktisch zu 100% gegen eine versehentliche Ausführung abgesichert werden.

Verfahren zum Verhindern der versehentlichen Veränderung von Speicherinhalten, insbesondere von Flash EEPROM-Speichern, sind verbreitet.

Aus der DE 196 16 053 A1 ist ein Verfahren zum Betreiben eines Steuergeräts mit einer programmierbaren Speichereinrichtung bekannt. Die Programmierung der Speichereinrichtung erfolgt dabei unter aufeinanderfolgender Ausführung einer Vielzahl von Speicherprogrammierungs-Steuervorgängen.

Dabei soll erreicht werden, dass ein durch Störungen ausgelöstes und/oder beeinflusstes Löschen und/oder Überschreiben von in der programmierbaren Speichereinrichtung gespeicherten Daten auf einfache Weise weitestgehend auszuschließen ist.

Dies wird dadurch erreicht, dass ein Überprüfungsschritt, durch welchen feststellbar ist, ob alle ausgewählten einzelnen oder mehreren der bis dahin auszuführenden Speicherprogrammierungs-Steuervorgänge ausgeführt wurden,

und ein Entscheidungsschritt, in welchem unter Berücksichtigung des Überprüfungsergebnisses entschieden wird, ob der Programmiervorgang unter Ausführung weiterer Speicherprogrammier-Steuervorgänge bestimmungsgemäß fortzusetzen ist, vorgesehen sind.

So ist es möglich, zu jedem beliebigen Zeitpunkt festzustellen, ob die bis dahin auszuführenden Steuervorgänge auch tatsächlich ausgeführt wurden.

Die EP 0 923 081 A2 beschreibt ein Verfahren zum Beschreiben und/oder Löschen eines Flash EEPROMs.

Bei diesem entspricht eine Programmier- bzw. Löschspannung einer Lesespannung. Zum Unterscheiden der Programmier- bzw. Löschspannung von der Lesespannung sind die Programmier- bzw. Löschalgorithmen mit definierten Adressen und definierten Daten in einer definierten Reihenfolge abzuarbeiten. Die Programmier- bzw. Löschalgorithmen sind in einem dem Flash EEPROM logisch zugeordneten flüchtigen Speicher verlagert.

Dadurch wird eine erhöhte Sicherheit der in dem Flash EEPROM abgelegten Speicherinformationen gegenüber zufälligem Überschreiben bzw. Löschen erreicht.

Eine durch zufälliges Beaufschlagen des Flash EEPROMs mit einer elektromagnetischen Entladung, durch Programmierfehler, Hardwarefehler und/oder Spannungspulse verursachte versehentliche Überschreibung oder Löschung gespeicherten Inhalts wird vermieden.

Weiterhin ist eine Ausführungsform beschrieben, bei der die für die Abarbeitung der Programmier- bzw. Löschalgorithmen

notwendigen Adressinformationen und Dateninformationen erst durch ein externes Programmiergerät bereitgestellt werden.

Vergleichbare Verfahren zur Absicherung beliebiger sicherheitskritischer Programmteile sind hingegen noch nicht bekannt. Hier setzt die Erfindung an.

Vorteile der Erfindung

Das erfindungsgemäße Verfahren dient zur Absicherung sicherheitskritischer Programmteile vor versehentlicher Ausführung. Bei diesem wird ein Programm mit mindestens einem Programmteil mit vorgegebener zeitlicher Abfolge ausgeführt. Zu einem bestimmten Zeitpunkt der Ausführung wird ein Muster erzeugt und zu mindestens einem späteren Zeitpunkt wird geprüft, ob das Muster vorhanden ist. Bei Fehlen des Musters wird die Ausführung des betreffenden Programmteils abgebrochen.

Als Muster kann ein einzelnes Bit, welches gesetzt wird, oder auch ein Bitmuster verwendet werden.

Bevorzugt wird das Muster in einem flüchtigen Speicher, z.B. einem RAM-Baustein, erzeugt. So ist sichergestellt, dass nach einem Programmstart ein bei einem früheren Programmdurchlauf erzeugtes Muster wieder gelöscht ist.

Wichtig beim erfindungsgemäßen Verfahren ist, dass bei der Ausführung sicherheitskritischer Programmteile zumindest zu einem Zeitpunkt überprüft wird, ob ein Muster vorhanden ist. Bei dessen Fehlen, weil beispielsweise aufgrund eines Fehlers der Programmzeiger in den sicherheitskritischen Bereich gesprungen ist, wird die Ausführung des Programmteils abgebrochen.

Die erfindungsgemäße Speichereinrichtung zur Ausführung des erfindungsgemäßen Verfahrens ist in ein oder mehrere Bereiche aufgeteilt. Dabei ist in jedem Bereich ein Programmteil abgelegt. Der bzw. die Programmteile sind mit einem vorgegebenen zeitlichen Ablauf ausführbar. Die erfindungsgemäße Speichereinrichtung zeichnet sich dadurch aus, dass ein Mittel vorgesehen ist, welches bewirkt, dass bei der Ausführung der Programmteile zu einem bestimmten Zeitpunkt ein Muster erzeugt wird, und wenigstens ein weiteres Mittel vorgesehen ist, welches bewirkt, dass zu einem späteren Zeitpunkt überprüft wird, ob das Muster vorhanden ist.

Das in der Speichereinrichtung abgelegte Programm wird dabei mittels eines Mikroprozessors abgearbeitet.

Es ist von Vorteil, die Mustererzeugung zu einem möglichst frühen Zeitpunkt durchzuführen.

Außerdem kann in der Speichereinrichtung eine Programmroutine vorgesehen sein, die ein Zurücksetzen des Prozessors (Reset) bewirkt. Bei einem fehlerfreien Durchlauf des Programms überspringt der Programmzeiger diese Programmroutine. Springt der Programmzeiger aber aufgrund eines Fehlers in die Programmroutine, wird der Prozessor zurückgesetzt.

Als Speichereinrichtung zur Ausführung des erfindungsgemäßen Verfahrens werden zweckmäßigerweise geeignete Datenträger, wie EEPROMs, Flash Memories, aber auch CD-ROMs, Disketten oder Festplattenlaufwerke verwendet.

Zeichnungen

Die Erfindung wird nachfolgend anhand der beigefügten Zeichnung und unter Bezugnahme auf ein Ausführungsbeispiel näher erläutert. In der Zeichnung zeigt:

Figur 1 eine herkömmliche Speichereinrichtung in schematischer Darstellung,

Figur 2 eine bevorzugte Ausführungsform der erfindungsgemäßen Speichereinrichtung in schematischer Darstellung, und

Figur 3 den Ablauf einer bevorzugten Ausführungsform des erfindungsgemäßen Verfahrens im Flussdiagramm.

In Figur 1 ist in schematischer Darstellung eine herkömmliche Speichereinrichtung 10 dargestellt. Die Speichereinrichtung 10 weist einen ersten Bereich 11 und einen zweiten Bereich 12 auf. Ein Pfeil 13 zeigt die Ausführungsrichtung an. Im ersten Bereich 11 ist ein erster Programmteil 14 und im zweiten Bereich 12 ein zweiter Programmteil 15 enthalten. Außerdem sind im zweiten Bereich 12 Daten in einem Datenfeld 16 abgelegt.

Der in der dargestellten Speichereinrichtung 10 vorgesehene zweite Programmteil 15 darf zu einem bestimmten Zeitpunkt (d. h. in einem bestimmten Betriebszustand) keinesfalls ausgeführt werden, da in diesem sicherheitskritische Routinen enthalten sind.

Der erste Programmteil 14 und der zweite Programmteil 15 sind so ausgelegt, daß der zweite Programmteil 15 direkt

vom ersten Programmteil 14 aufgerufen wird oder zumindest eine feste kausale Verknüpfung zwischen den beiden besteht.

In Figur 1 sind ein erster kritischer Bereich 17 und ein zweiter kritischer Bereich 18 kenntlich gemacht. Die kritischen Bereiche 17, 18 umfassen die Teile des Speichers, in welchen ein versehentliches Einspringen des Programmzeigers zu einer vollständigen Ausführung des sicherheitskritischen zweiten Programmteils 15 führen kann.

Gründe für den Umfang des kritischen Bereichs sind:

1. Ein Einsprung des Programmzeigers in den ersten Programmteil 14 führt direkt oder indirekt zu einer weiteren Ausführung des zweiten Programmteils 15 und damit der sicherheitskritischen Routinen.

2. Ein Einsprung des Programmzeigers in das Datenfeld 16 führt zunächst zu einer Ausführung unsinniger Befehle (Daten als Opcode),welche aber nicht notwendigerweise zu einem Rücksetzen des Prozessors (Reset) führen müssen. Es besteht daher die Möglichkeit, dass der Programmzeiger bis in den Bereich des ausführbaren zweiten Programmteils 15 weiterläuft und damit die sicherheitskritischen Routinen zur Ausführung kommen.

3. Der direkte Einsprung des Programmzeigers in den zweiten Programmteil 15 führt zwangsläufig zu einer kompletten Ausführung der sicherheitskritischen Routinen. Dies kann hier beispielsweise zu einer Ausführung einer Endlosschleife führen. Um dies zu verhindern, wird erfindungsgemäß folgendermaßen vorgegangen.

Ausführungsbeispiel

In Figur 2 ist eine erfindungsgemäße Speichereinrichtung mit der Bezugsziffer 20 schematisch dargestellt.

Die erfindungsgemäße Speichereinrichtung 20 weist einen ersten Bereich 21 und einen zweiten Bereich 22 auf. Ein Pfeil 23 verdeutlicht die Ausführungsrichtung. Im ersten Bereich 21 ist ein erster Programmteil 24 und im zweiten Bereich 22 ein zweiter Programmteil 25 vorgesehen. Der zweite Programmteil enthält wiederum sicherheitskritische Routinen.

Weiterhin umfaßt der zweite Bereich 22 ein Datenfeld 26.

Im ersten Bereich 21 ist ein erstes Mittel 27 vorgesehen, welches bewirkt, dass ein Muster erzeugt wird. Das erste Mittel 27 ist beispielsweise eine Programmroutine, welche ein Bit oder ein Bitmuster setzt.

Entsprechend ist im zweiten Bereich 22 ein Mittel 28 vorgesehen, welches eine Musterprüfung durchführt. Das zweite Mittel 28 ist z.B. eine Programmroutine, die überprüft, ob das entsprechende Bit bzw. das Bitmuster gesetzt ist.

Zusätzlich ist im zweiten Bereich 22 eine Programmroutine 29 dargestellt, welche ein Zurücksetzen des Prozessors (Reset) bewirkt.

Bei der erfindungsgemäßen Speichereinrichtung 20 sind somit, wie nachfolgend erläutert wird, Maßnahmen getroffen, um die versehentliche Ausführung des sicherheitskritischen zweiten Programmteils 25 zu vermeiden.

Zunächst wird zu Beginn des ersten Programmteils 24 durch das erste Mittel 27 ein Muster im flüchtigen RAM erzeugt, welches für die Ausführung des zweiten Programmteils 25 zwingend erforderlich ist. Ein Einsprung des Programmzeigers in den ersten Bereich 21 hinter dem ersten Mittel 27 führt zunächst direkt oder indirekt zu einer weiteren Ausführung des zweiten Programmteils 25. Die Ausführung des zweiten Programmteils 25 wird jedoch aufgrund des fehlenden Musters, was durch die Überprüfung mit dem Mittel 28 festgestellt wird, abgebrochen. Eine versehentliche Erzeugung des Musters ist nur möglich, wenn der Programmzeiger vor die Ausführung der Erzeugung des Musters durch das erste Mittel 27 in den Programmablauf einspringt.

Hieraus wird deutlich, dass das erste Mittel 27 das Muster bevorzugt zu einem möglichst frühen Zeitpunkt in der Programmausführung erzeugt.

Ferner wird am physikalischen Beginn des zweiten Bereiches 22, in welchem sich der sicherheitskritische zweite Programmteil 25 befindet, der Befehl zum Zurücksetzen des Prozessors (Reset) hinterlegt. Ein Einsprung des Programmzeigers in das Datenfeld 26 führt zunächst wie im ungeschützten Fall zu einer Ausführung unsinniger Befehle (Daten als Opcode), welche aber nicht notwendigerweise zu einem Zurücksetzen des Prozessors (Reset) führen müssen. Es besteht daher ebenfalls die Möglichkeit, dass der Programmzeiger weiterläuft. Bei Erreichen der Programmroutine 29 wird dann zwangsläufig der Prozessor zurückgesetzt und eine weitere Ausführung des sicherheitskritschen zweiten Programmteils 25 vermieden.

Bei einem fehlerfreien Ablauf des Programms springt der Programmzeiger aus dem ersten Bereich 21 in den zweiten Bereich 22 hinter die Programmroutine 29.

Weiterhin führt der direkte Einsprung des Programmzeigers in den zweiten Bereich 22 zwangsläufig zu einer Ausführung des nachfolgenden, sicherheitskritischen zweiten Programmteils 25 bis zur nächsten hinterlegten Prüfung des erforderlichen Musters. Da kein Muster an der Stelle 27 im Bereich 21 erzeugt wurde, wird die Ausführung des sicherheitskritischen zweiten Programmteils 25 an dieser Stelle abgebrochen.

Die Felder 30 und 31 verdeutlichen den kritischen Bereich, der sich im Vergleich zu den kritischen Feldern 17 und 18 aus Figur 1 erheblich verringert hat. Die Felder 32 und 33 zeigen den geschützten Bereich, das Feld 34 den gegen wiederholte Ausführung geschützten Bereich.

Werden bei der Musterprüfung und der Mustererzeugung jeweils externe Randbedingungen (z. B. Zustände von Hardwarekomponenten) geprüft, so kann die versehentliche Ausführung des zweiten Programmteils 25 über einen Einsprung in den ersten Bereich 21 sogar zu 100% verhindert werden.

Soll beispielsweise die Ausführung des ersten Programmteils 24 und des zweiten Programmteils 25 von einer äußeren Bedingung abhängen, so wird zusätzlich diese Bedingung, beispielsweise ein Spannungswert, zusätzlich noch einmal überprüft.

Figur 3 verdeutlicht den Ablauf des erfindungsgemäßen Verfahrens im Flussdiagramm.

Mit einem Schritt 40 beginnt die Ausführung des Programms. Der Programmzeiger springt in den ersten Bereich 27.

In einem Schritt 41 wird ein Muster erzeugt. Der im ersten Bereich 21 abgelegte erste Programmteil 24 kommt zur Ausführung. Nach Beendigung dieses Programmteils springt der Programmzeiger mit einem Schritt 42 in den zweiten Bereich 22.

Im zweiten Bereich 22 wird in einem Schritt 43 der zweite Programmteil 25 ausgeführt und zusätzlich zu unterschiedlichen Zeitpunkten überprüft, ob das im ersten Bereich 21 erzeugte Muster im RAM abgelegt ist. Ist dies nicht der Fall, erfolgt der Abbruch des Programms 44. Liegt ein Muster vor, wie dies bei einem fehlerfreien Ablauf des Programms der Fall ist, wird das Programm in einem Schritt 45 weiter ausgeführt.

Mit Hilfe des erfindungsgemäßen Verfahrens kann die Wahrscheinlichkeit der versehentlichen Ausführung der sicherheitskritischen Programmteile und das daraus resultierende Fehlverhalten des Steuergeräts in erheblichem Maße, d. h. um Größenordnungen, reduziert werden.

Das beschriebene Verfahren eignet sich vor allem zur Vermeidung der Ausführung von sicherheitskritischen, zyklisch ausgeführten Programmteilen. Eine zyklische Ausführung kann durch die Hinterlegung einer einzigen Prüfung des relevanten Musters verhindert werden.

Programmteile, welche auch bei einmaliger Ausführung sicherheitskritisch sind, können durch mehrfache Prüfung in möglichst kleine Teile zerlegt werden.

28.02.01

ROBERT BOSCH GMBH, 70442 Stuttgart

Ansprüche

1. Verfahren zur Absicherung sicherheitskritischer Programmteile vor versehentlicher Ausführung, bei dem mindestens ein Programmteil mit vorgegebenem zeitlichen Ablauf ausgeführt wird,
d a d u r c h g e k e n n z e i c h n e t ,
dass zu einem bestimmten Zeitpunkt der Ausführung ein Muster erzeugt wird und anschließend zumindest zu einem Zeitpunkt geprüft wird, ob das Muster vorhanden ist.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass das Muster zu Beginn der Programmausführung erzeugt wird.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, dass das Muster in einem flüchtigen Speicherelement erzeugt wird.

4. Verfahren nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, dass bei der Mustererzeugung und bei der Musterprüfung jeweils externe Randbedingungen geprüft werden.

5. Verfahren nach Anspruch 4, dadurch gekennzeichnet, dass als externe Randbedingungen Zustände von Hardwarekomponenten dienen.

6. Verfahren zur Absicherung von sicherheitskritischen Programmteilen, bei dem bei der Ausführung eines sicherheitskritischen Programmteils zumindest zu einem Zeitpunkt überprüft wird, ob ein den ordnungsgemäßen Ablauf des Programms repräsentierendes Muster vorhanden ist, und bei dessen Fehlen die Ausführung des Programmteils abgebrochen wird.

7. Speichereinrichtung zur Ausführung eines Verfahrens nach einem der Ansprüche 1 bis 6 mittels eines Mikroprozessors, welche in einen oder mehrere Bereiche (21, 22) aufgeteilt ist, wobei in jedem Bereich ein Programmteil angelegt ist, und der mindestens eine Programmteil mit vorgegebenem zeitlichen Ablauf ausführbar ist, dadurch gekennzeichnet, daß ein erstes Mittel (27) vorgesehen ist, welches bewirkt, dass bei der Ausführung der Programmteile zu einem bestimmten Zeitpunkt ein Muster erzeugt wird und wenigstens ein weiteres zweites Mittel (28) vorgesehen ist, welches bewirkt, dass zu einem späteren Zeitpunkt überprüft wird, ob das Muster vorhanden ist.

8. Speichereinrichtung nach Anspruch 7, gekennzeichnet durch ein Mittel, das ein Zurücksetzen des Prozessors bewirkt.

28.02.01

ROBERT BOSCH GMBH, 70442 Stuttgart

Verfahren zur Absicherung sicherheitskritischer Programmteile vor versehentlicher Ausführung und eine Speichereinrichtung zur Durchführung dieses Verfahrens

Zusammenfassung

Es wird ein Verfahren zur Absicherung sicherheitskritischer Programmteile vor versehentlicher Ausführung beschrieben.

Bei diesem wird mindestens ein Programmteil mit vorgegebenem zeitlichen Ablauf ausgeführt. Zu einem bestimmten Zeitpunkt der Ausführung wird ein Muster erzeugt und zu mindestens einem späteren Zeitpunkt geprüft, ob das Muster vorhanden ist, und bei Fehlen des Musters die Ausführung des betreffenden Programmteils abgebrochen.

Weiterhin wird eine Speichereinrichtung (20) zur Ausführung eines solchen Verfahrens beschrieben.

(Fig. 2)

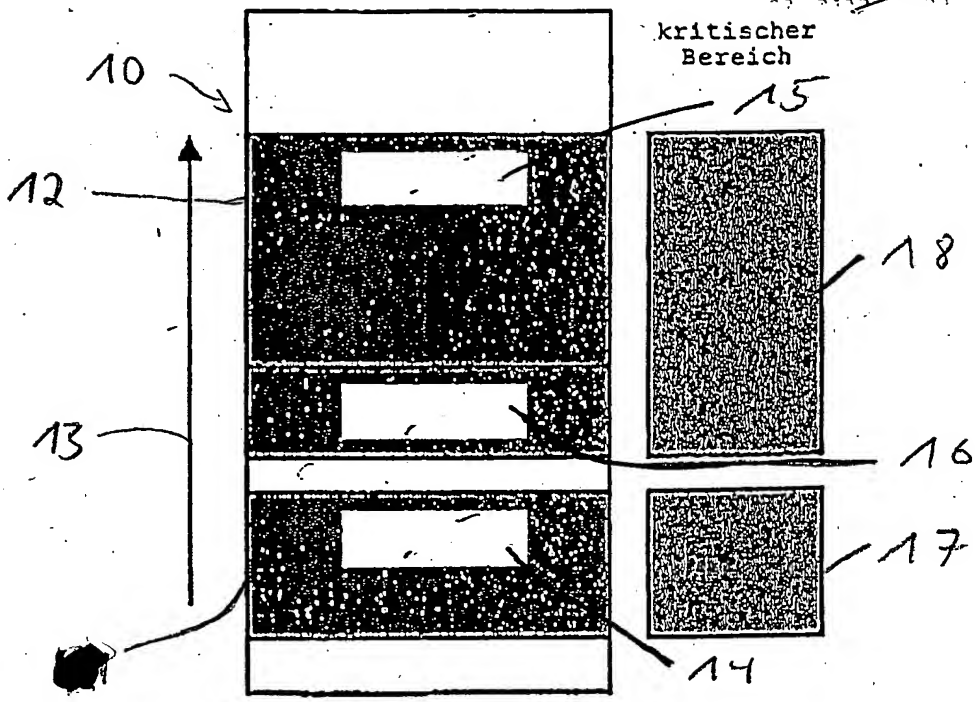

Fig. 1

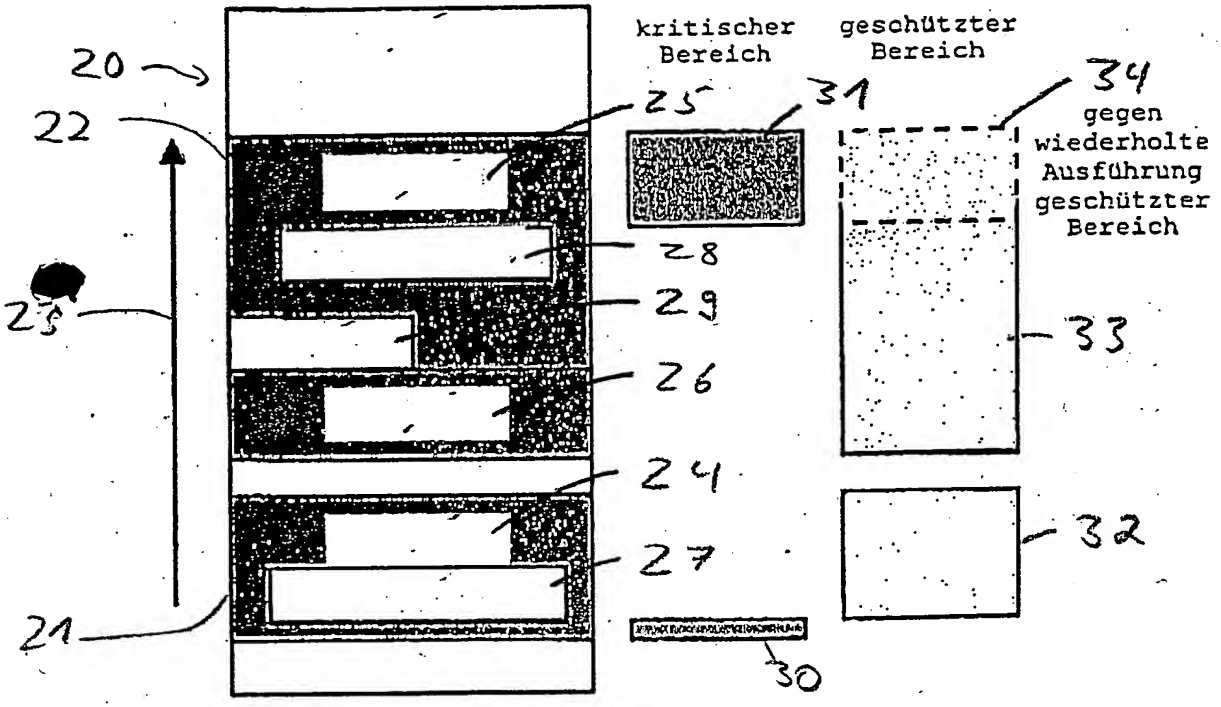

Fig. 2

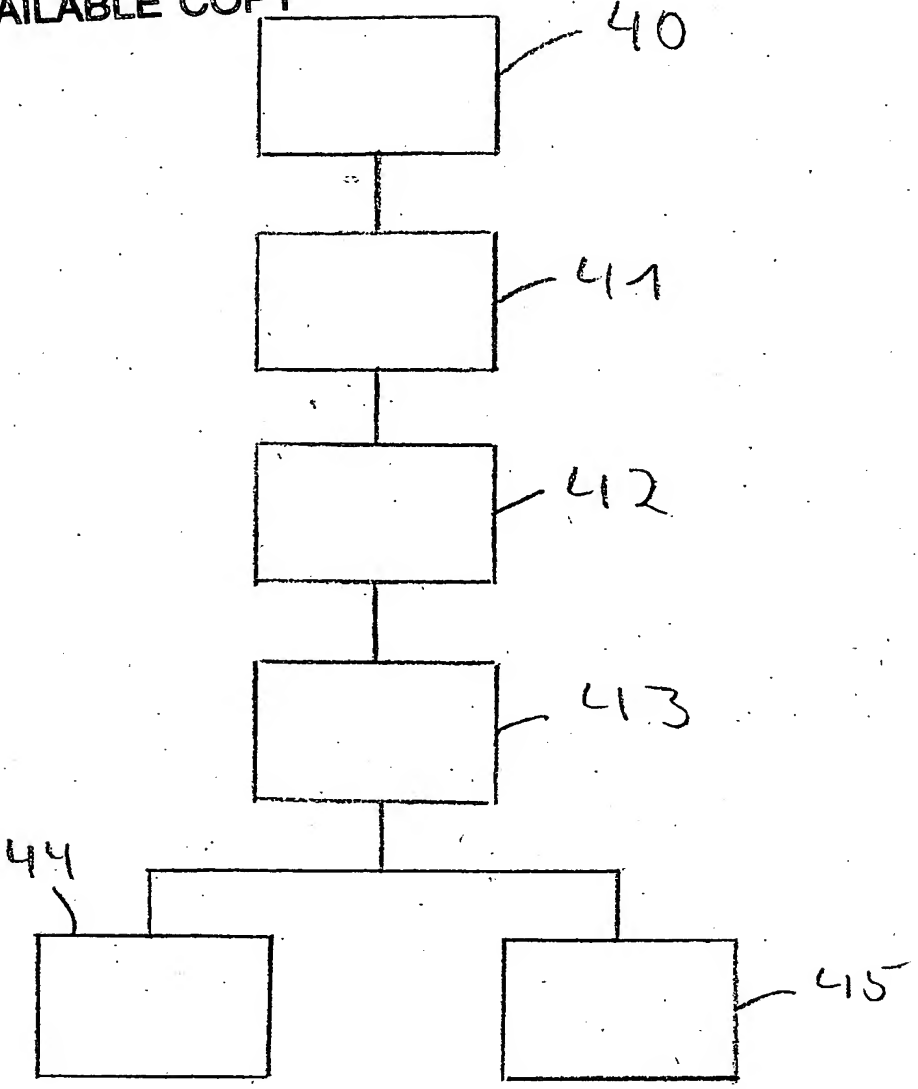

Fig. 3